## 0.1 Graph Grammars

A *Graph Grammar* (GG) is composed of a set of graph transformation rules. These rules indicate how a graph can be transformed to a new graph. These graphs are called *host graphs*. The rules are composed of graphs themselves, which are called *rule graphs*.

The rest of this section is ordered as follows: first, graphs, host graphs, rule graphs and graph transformation rules are explained. Then, the definition of a *Graph Transition System* (GTS) is given. An example of a GG and a GTS is then given. Finally, the definition of IOGGs is given. For a more detailed overview of GGs, we refer to [**?**, **?**, **?**].

**Definition 0.1.1.** A *graph* is composed of nodes and edges. In this report, we assume a universe of nodes $\mathbb{V} = \mathbb{W} \uplus \mathbb{U} \uplus \mathcal{V} \uplus 2^{\mathcal{T}}$, where $\mathbb{W}$ is the universe of standard graph nodes. $\mathbb{E}$ is the universe of edges between two nodes in $\mathbb{V}$.

**Definition 0.1.2.** A host graph $G$ is a tuple $\langle V_{G^h}, E_{G^h} \rangle$, where:

- $V_{G^h} \subseteq (\mathbb{W} \uplus \mathbb{U})$ is the node set of $G$
- $E_{G^h} \subseteq (V_{G^h} \backslash \mathbb{U} \times L \times V_{G^h})$ is the edge set of $G$

Figure 1 shows an example of a host graph. Here, $n_1, n_2 \in \mathbb{W}$ are the *identities* of the nodes. The other four nodes are values in $\mathbb{U}$.

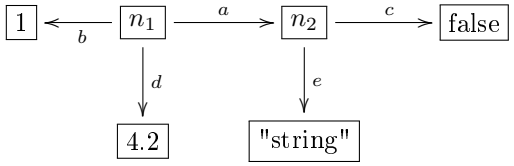



Figure 1: An example of a host graph

**Definition 0.1.3.** A rule graph $H$ is a tuple $\langle V_{G^r}, E_{G^r} \rangle$, where:

- $V_{G^r} \subseteq (\mathbb{V} \backslash \mathbb{U})$ is the node set of $H$
- $E_{G^r} \subseteq (V_{G^r} \times L \times V_{G^r})$ is the edge set of $H$

In addition, the following must hold:

- $\forall z \in V_{G^r} \wedge z \in 2^{\mathcal{T}} \,.\, var(z) \subseteq V_{G^r}$ - The variables used in the terms must be present as nodes in the rule graph.
- $(\forall z \in V_{G^r} \wedge z \in \mathcal{V} \,.\, \exists(\_, \_, z) \in E_{G^r}$ - If a variable is used in a rule graph, it needs context. Therefore, there must be an edge with the variable node as target.

Figure 2 shows an example of a rule graph. Here, $r_1, r_2 \in \mathbb{W}$ are the node identities, $x_1, x_2 \in \mathcal{V}^{int}$ and $\{x_1 + 1, x_2\} \in 2^{\mathcal{T}}$. The set of terms is mapped as a node to the same value. This mapping is explained in the next definition. The consequence is that this node implicitly expresses the relation $x_1 + 1 = x_2$.

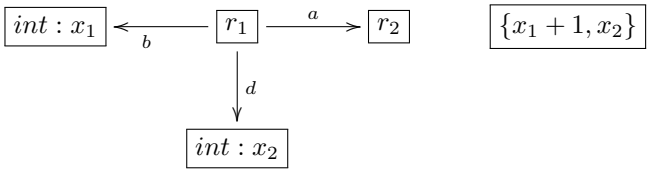



Figure 2: An example of a rule graph

**Definition 0.1.4.** A graph $g$ has a *morphism* to a graph $g'$ if there is a structure-preserving mapping from the nodes and the edges of $g$ to the nodes and the edges of $g'$ respectively. A graph $g$ has a partial morphism to a graph $g'$ if there are elements in $g$ without an image in $g'$.

**Definition 0.1.5.** A node or edge $z$ in graph $g$ has an *image* in graph $g'$ and $z$ is a *pre-image* of the image. For each type of node, an explanation is given A variable $v \in \mathcal{V}^s, s \in S$, has an image $i$ in a host graph if $i \in \mathbb{U}^s$. A node $z \in 2^{\mathcal{T}}$ has an image $i$ in a host graph if $i$ is the valuation of all terms in $z$.

**Definition 0.1.6.** A transformation rule is a tuple $\langle LHS, NAC, RHS, l \rangle$, where:

- $LHS$ is a rule graph representing the left-hand side of the rule
- $NAC$ is a set of rule graphs representing the negative application conditions
- $RHS$ is a rule graph representing the right-hand side of the rule
- $l \in L$ is the label of the rule

There exist implicit partial morphisms from the $LHS$ to each rule graph in $NAC$ and from the $LHS$ to the $RHS$ by means of the node identities. These morphisms are *rule graph morphisms*.

**Definition 0.1.7.** A rule $r$ has a *rule match* on a host graph $G$ if its $LHS$ has a morphism in $G$ and $\nexists n \in NAC$ such that $n$ has a morphism in $G$ and $\forall e \in LHS$, if $e$ has an image $i$ in $n$, and an image $j$ in $G$, then $j$ should be an image of $i$. The morphism of the $LHS$ to a host graph is a *match morphism*.

**Definition 0.1.8.** After the rule match is applied to the graph, all elements in $LHS$ that do not have an image in $RHS$, are removed from $G$ and all elements in $RHS$ that do not have a pre-image in $LHS$, are added to $G$. This process, called a *rule transition*, is denoted $G \xrightarrow{r,m} G'$, where $m \in M$ is the morphism of the $LHS$ to $G$.

Figure 3 shows an example of the initial graph $G_0$, one rule of a GG and the corresponding rule match. $G_0$ can be represented by $\langle \{n1, n2\}, \{\langle n1, a, n1\rangle, \langle n1, A, n2\rangle, \langle n2, B, n2\rangle\}\rangle$. The $LHS$ of the rule has a match in $G_0$. Neither $NAC1$ and $NAC2$ have a match in $G_0$, because the edge with label $C$ does not exist in $G_0$. The new graph after applying the rule is $G_1$.

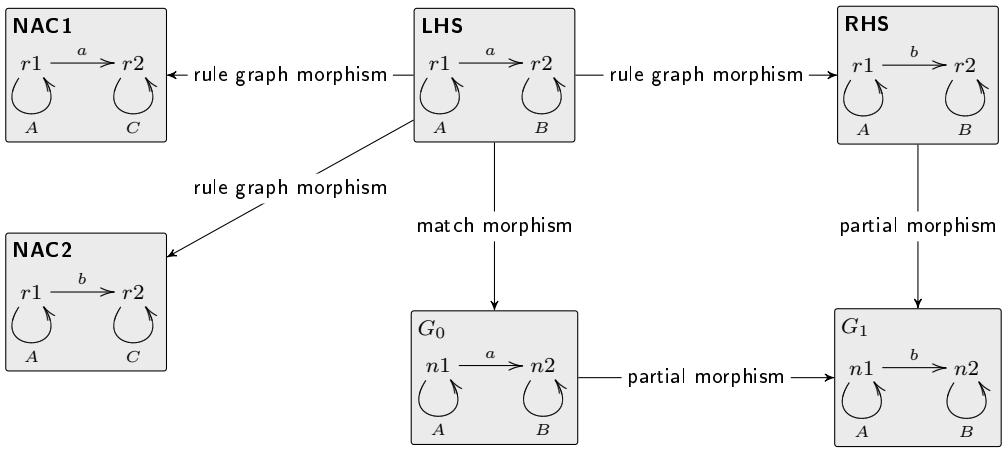



Figure 3: An example of a graph transformation

**Definition 0.1.9.** A graph grammar is a tuple $\langle R, G_0 \rangle$, where:

- $R$ is a set of graph transformation rules
- $G_0$ is the initial graph

By repeatedly applying graph transformation rules to the start graph and all its consecutive graphs, a GG can be explored to reveal a *Graph Transition System* (GTS). This transition system consists of graphs connected by rule transitions.

**Definition 0.1.10.** A graph transition system is a tuple $\langle \mathcal{G}, R, M, U, G_0 \rangle$, where:

- $\mathcal{G}$ is a set of graphs
- $L \in R \times M$ is a set of labels
- $U \in \mathcal{G} \times L \times \mathcal{G}$ is the rule transition relation
- $G_0 \in \mathcal{G}$ is the initial graph

Let $K = \langle R, G_0$. A GTS $O = \langle \mathcal{G}, R, M, U \rangle$ is derived from a $K$ by the following. $\mathcal{G}, M, U$ are the smallest sets, such that:

- $G_0 \in \mathcal{G}$
- if $G \in \mathcal{G}$ and $G \xrightarrow{r,m} G'$ then $G' \in \mathcal{G}, (r, m) \in L, (G \xrightarrow{r,m} G') \in U$

**Definition 0.1.11.** In order to specify stimuli and responses with GGs, a definition is given for an *Input-Output GG* (IOGG). Concretely, the IOGG places input and output labels on its rule transitions. Following the definition from IOLTSs, each rule label $l \in L$ has a type $\iota \in Y$. Exploring an IOGG leads to an *Input-Output Graph Transition System* (IOGTS). The rule transitions derive their type from their corresponding rule.

# Chapter 1

# From Graph Grammar to STS

## 1.1 Requirements considerations

In order to do model-based testing with GGs, stimuli and responses have to be obtained from the GG. ATM uses an IOSTS, where the instantiated switch relations represent a stimulus to or a response from the SUT. To get an equivalent notion of stimuli/respones in GGs, the GG must be extended to an IOGG by indicating for each transformation rule whether it is of the input or output type. Then the IOGG can be explored to an IOGTS. The input/output rule transitions of the IOGTS can be used as the abstract stimuli and responses.

The second requirement for the design is the possibility to measure coverage statistics. The exploration of a GG can be done in two ways: *on the fly*, rule transitions are explored only when chosen by ATM, or *offline*, the GG is first completely explored and then sent to ATM. On-the-fly model exploration works well on large and even infinite models. However, coverage statistics cannot be calculated with this technique. The number of states (graphs) and rule transitions the model has when completely explored are not known, so a percentage cannot be derived. As coverage statistics are an important metric, the offline model exploration is chosen for GRATiS.

The last requirement is efficiency. An IOGTS can potentially be infinitely large, due to the range of data values. A model that is more efficient with data values is an STS. The setup of GRATiS is therefore to transform the IOGG directly to an IOSTS. Note that the first requirement is met, because locaton and switch relation coverage can be calculated on the IOSTS.

Taking these requirements into account, the method to achieve the goal of model-based testing on GGs is the following three steps:

1. Assign I/O types to graph transformation rules
2. Create an IOSTS from the IOGG
3. Perform the model-based testing on the IOSTS

This chapter describes an algorithm for creating an IOSTS from an IOGG.

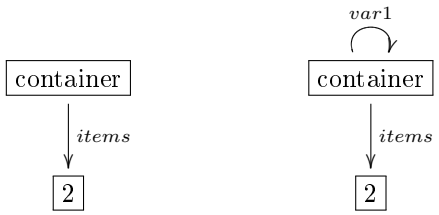
## 1.2 Point algebra

We define a *point algebra* $\mathcal{P}$ to be an algebra with $\forall s \in S.|\mathbb{U}^s_{\mathcal{P}}| = 1$. Each graph in $\mathcal{G}$ using the point algebra is structurally unique upto isomorphism; different values on value nodes are eliminated by the point algebra and two structurally equivalent graphs are the same graph. Therefore, using this

algebra is efficient when exploring the GTS. The loss of information is only in the concrete values at each state. This information is also not present in an STS, which treats the values symbolically as variables.

## 1.3 Variables

The variables in an STS represent an aspect of the modelled system. For instance, if a system keeps track of the number of items in containers, the STS modelling this system could have integer location variables $items_1..items_n$. The value nodes in a host graph are a representation of one element from the universe of elements of the same sort. Edges can exist between graph nodes and value nodes. The same example modelled in a graph grammar could be a graph node representing a container with an edge labelled 'items' to an integer node. This is shown in Figure 1.1a. This is a common way of representing a variable in a GG. Here the combination of edge plus source node represents the variable. However, the source node identity is not consistent through graph transformations, as the graphs are structurally unique upto isomorphism. In order to have variables in GGs, the source node must be made structurally unique, by means of a self-edge. Figure 1.1b shows the self-edge on the container node. The variable *var1_items*, the number of items in the container, is now represented by this graph.




(a) Common way (b) Persistent way

Figure 1.1: Possible ways of modelling variables in GGs

On the basis of the discussion above, we introduce the following terminology. The labels on the self-edges we call *variable labels*, represented by $L_{var}$. The edge having a variable label we call a *variable edge*. The source and target node of the variable edge we call the *variable anchor*.

## 1.4 The algorithm

Let $J = \langle W, w_0, \mathcal{L}, \imath, \mathcal{I}, \Lambda, D \rangle$ be an IOSTS and let $K = \langle R, G_0 \rangle$ be an IOGG. The first step in the algorithm is to explore the GG $K$ using the point algebra $\mathcal{P}$ to an IOGTS $O_{\mathcal{P}} : \langle G, R, M, U, G_0 \rangle$.

### 1.4.1 Locations

The set of locations $W$ is chosen to be equal to the set of graphs $G$. Additionally, the initial location $w_0$ is equal to the initial graph $G_0$.

### 1.4.2 Location variables

The location variables are a subset of the product of variable labels and regular labels, given by $\mathcal{L} \subset L_{var} \times L$. The set of location variables is defined by the following. $\langle l_{var}, l \rangle \in \mathcal{L}$ if:

- $\langle w \in \mathbb{W}, l, u \in \mathbb{U} \rangle \in \mathbb{E}$ - the label must be on an edge from a graph node to a value node.
- $\langle w, l_{var}, w \rangle \in \mathbb{E}$ - the variable label must be a self edge on the same graph node.

The initialization $\imath$ is then given by $\langle l_{var}, l \rangle \mapsto u$.

### 1.4.3 Gates

The gate of a switch relation represents the stimulus to or response from the SUT. In an IOGG, the rules are this representation. Therefore, the set of gates $\Lambda$ is chosen to be equal to the set of rules $R$.

### 1.4.4 Interaction variables

Interaction variables are used by the gates to represent a stimulus or response variable. The variable nodes in rule graphs are this representation. The set of interaction variables $\mathcal{I}$ is chosen to be equal to the set of variable nodes $\mathcal{V}$. For a rule $r$ and all variable nodes $\mathcal{V}_r$ in $LHS$ of $r$, $arity(r) = |\mathcal{V}_r|$.

### 1.4.5 Guards

The guard of a switch relation restricts the use of the switch relation based on the values of the variables. In a GG, a rule is restricted by the terms. The variables used in the terms are interaction variables. Therefore, the first part of the guard is constructed by joining the terms for each term node by $\bigwedge_{z \in V_{G^r} \cap 2^{\mathcal{T}} || z | > 1} \bigwedge_{t_1, t_2 \in z} t_1 = t_2$. Using a rule match $m$, the second part is constructed. For a $LHS = \langle V_{G^r}, E_{G^r} \rangle$ the smallest set of terms $T$, such that $\langle m(z), l \rangle = x \in T$ when:

- $x \in \mathcal{V} \cap V_{G^r}$
- $\langle z, l, x \rangle \in E_{G^r}$
- $m(z)$ is a variable anchor

Then, the terms are joined by $\bigwedge_{t_1, t_2 \in T} t_1 \wedge t_2$.

### 1.4.6 Update mappings

An edge with label $l$ from a variable anchor $z$ to a value node can be erased from the graph and a new edge with label $l$ from $z$ to a new value node can be created by a rule. This inidicates an update for the location variable given by $\langle z, l \rangle$. In the rule graph, the $RHS$ of the rule has the pre-image of the $z$ and the edge to a variable node or term node, given by the interaction variable $x$. The update mapping for this example is: $\langle z, l \rangle \mapsto x$.

### 1.4.7 Switch relations

A rule transition $G \xrightarrow{r,m} G' \in U$ is mapped to a switch relation $(G \xrightarrow{r,\gamma,\rho} G') \in D$. The guard and update mapping are constructed according to sections 1.4.5 and 1.4.6 using $r$ and $m$.